Journal of Multimedia Information System VOL. 12, NO. 3, September 2025 (pp. 135-139): ISSN 2383-7632 (Online)
https://doi.org/10.33851/JMIS.2025.12.3.135

glTF Extension Format with LOD and WebGL 3D Visualization Platform

Jaeho Im[1*], Minsoo Park[1], Yujin Kim[1]

Abstract

This paper proposes an advanced web platform for rendering glTF models with Level of Detail (LOD) information, presenting an efficient technique for handling complex 3D mesh models containing extensive geometric and visual data. While glTF™ has emerged as a powerful API-neutral asset format optimized for efficient transmission and loading of 3D scenes and models, its original specification lacked native LOD support, creating significant limitations for web-based applications where bandwidth and processing capabilities vary widely across devices. Our research develops tools for generating glTF files containing EXT_lod tags based on Microsoft's proposed LOD extension technology, MSFT_LOD, and proposes techniques for rendering these files in web environments. We propose an appropriate LOD rendering technique for glTF viewer platforms by developing technology that controls LOD based on WebGL canvas size and memory usage. The system architecture supports progressive loading strategies, further enhancing user experience through reduced initial load times. Our technique contributes to efficient data management and personalization in the emerging decentralized 3D content ecosystem of Web3.0, providing a foundation for more accessible and resource-efficient 3D web applications.

Key Words: Web3.0, glTF, WebGL.

I. INTRODUCTION

Recent advances in computer vision technology and the proliferation of smartphones and drones have made high-resolution 3D scan data generation increasingly accessible. However, real-time rendering of these large-volume 3D models in web environments remains a significant technical challenge. glTF, a specification optimized for network transmission and loading of 3D models, supports rapid rendering through JSON and binary structures. While this format allows for feature additions through extension mechanisms, the standard specification is limited to handling only pure geometry-related information. Although custom extensions can incorporate additional information into glTF models, such data cannot be utilized by standard renderers that strictly adhere to the base specification. Large-scale 3D models, in particular, contain numerous vertex information points, making level of detail (LOD) data essential for efficient rendering. However, the lack of native LOD support in the glTF file format makes it difficult to deliver optimized rendering results across diverse platforms.

This paper proposes a WebGL-based platform technology utilizing glTF file formats that incorporate LOD information. Rather than developing an entirely new file format, we adopted the MSFT_LOD technology proposed by Microsoft, taking an approach that could gain recognition as a standard specification based on existing public information. MSFT_LOD incorporates LOD information into the glTF file format by adding EXT_lod elements using extension tags within the glTF file format. While MSFT_LOD is Windows-dependent and requires mandatory plugin installation, we developed importer and exporter technologies that can be easily utilized across web and various applications. Our research uses the HTML5 <canvas> element as the rendering surface and employs WebGL 2.0 (OpenGL ES 3.0 compatible) to render 3D models. WebGL-based visualization eliminates the need for plugins, enhancing user accessibility and enabling implementation across various platforms (PC, mobile, AR/VR). Furthermore, unlike conventional approaches that set LOD based on the distance between camera and object, we present a technique that selects LOD models based on either the canvas size within the web browser or memory usage, enabling effective LOD application even in viewers with

Manuscript received May 15, 2025; Revised June 17, 2025; Accepted June 19, 2025. (ID No. JMIS-25M-05-014)
Corresponding Author (*): Jaeho Im, +82-10-8881-6693, jaeho.im@dexterstudios.com
[1]R&D Department, Dexter Studios, Seoul, Korea, jaeho.im@dexterstudios.com, minsoo.park@dexterstudios.com, yujin.kim@dexterstudios.com

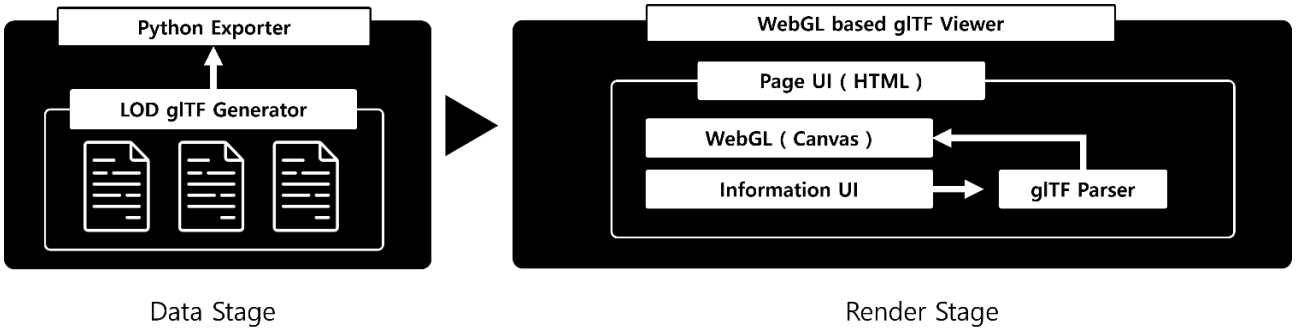


Fig. 1. System overview.

limited camera movement.

The technique we propose faithfully adheres to existing file formats and protocols while offering technology capable of efficiently processing and visualizing high-volume 3D data in the context of decentralized data management in the Web3.0 era, thereby presenting a paradigm for data visualization appropriate for the new digital age.

II. RELATED WORK

Level of detail (LOD) techniques are essential for efficient processing of high-resolution 3D scan data. In complex scenes, rendering load can be effectively reduced by selecting models with varying levels of detail based on camera distance or screen coverage. Research such as DECODE-3DViz has demonstrated that combining LOD with chunk streaming techniques can reduce rendering time of web-based volumetric data by up to 98% and significantly decrease GPU memory usage to just a few megabytes [1].

LOD support is not yet included in the glTF core specification, though Microsoft and others have proposed using separate extensions (MSFT_LOD) [2]. The Windows Mixed Reality home app guide recommends defining LOD levels and screen coverage through MSFT_LOD and MSFT_ScreenCoverage extensions [2]. This approach of adding LOD information to glTF through extension mechanisms is widely recommended in the industry.

In the field of WebGL visualization, numerous studies have explored level-of-detail control and streaming pipelines for efficient processing of large-scale data. For example, Cesium's 3D Tiles pipeline enables efficient streaming by converting high-resolution drone scan data into multi-resolution tiles [3].

The utilization of glTF is also being actively discussed in Web3.0-based 3D content areas. Platforms such as Web3DP implement distributed storage and management of 3D model assets by incorporating GLB/glTF file formats into NFT/ERC-721 standards [4]. These previous studies present possibilities for technical integration of glTF and WebGL and their application in Web3.0 environments.

Our research implements a WebGL and glTF-based adaptive LOD viewer building upon the achievements of these previous studies. However, unlike existing research, we propose an LOD selection algorithm based on browser canvas size and memory usage, offering an LOD application technique specifically suited for web browser-based viewer platforms. Additionally, while adopting Microsoft's MSFT_LOD extension, we optimized it for web environments, maintaining standard compatibility while tailoring it to the characteristics of web platforms. This provides a technical foundation that can contribute to 3D content distribution and efficient data management in the Web3.0 environment.

III. SYSTEM ARCHITECTURE AND ALGORITHMS

3.1. System Overview

The WebGL-based glTF multi-LOD rendering system proposed in this research consists of two main phases: the data generation phase and the LOD-based rendering phase (Fig. 1). This system provides a framework that enables efficient LOD-based 3D model rendering in a platform-independent environment while leveraging existing glTF extension capabilities.

3.2. LOD Integration Data Generation Technique

3.2.1. Python-Based Extensible glTF Exporter

Microsoft's MSFT plugin offers the advantage of effectively storing LOD information for multiple meshes; however, its dependency on mandatory installation of the MSFT Toolkit imposes limitations on its universal applicability. To overcome these constraints, this research has developed a Python-based glTF Exporter that incorporates LOD information.

The proposed Exporter takes N independent glTF files as input to generate a single glTF model, applying the

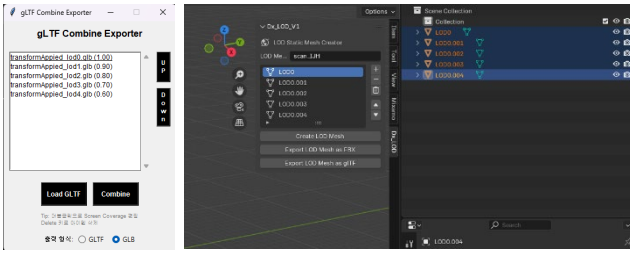


Fig. 2. Python-based standalone software for glTF integration (left) and application results within Blender environment (right).

EXT_Lod tag to maintain compatibility with the existing MSFT format. Implemented in a Python environment, it offers the advantage of immediate utilization across various platforms without requiring the installation of specialized software. In this research, we applied and utilized the developed converter within Blender (Fig. 2).

3.2.2. WebGL-Based Integrated Importer

The glTF files containing LOD information generated through the Exporter are loaded via the WebGL-based platform developed in this research. The Importer module parses the structured LOD data from the glTF file and dynamically generates vertex and index buffers based on this data for efficient rendering on the WebGL Canvas. This process maximizes rendering efficiency by selectively loading mesh data at appropriate LOD levels, utilizing information from the EXT_Lod tag.

3.3. Context-Adaptive LOD Transition Algorithm

In conventional 3D model viewers, LOD transitions are primarily determined based on the projection ratio within the screen, which depends on the distance between the camera and the target object. However, since our system is designed to maintain the camera within a certain distance from the object, the traditional distance-based LOD transition approach does not provide an optimal solution. Therefore, this research has developed the following three context-adaptive LOD transition algorithms.

3.3.1. Canvas Size-Based LOD Transition

This algorithm utilizes the rendering canvas size as a primary parameter for LOD selection. When the canvas size is dynamically adjusted in response to changes in the web browser size, according to HTML5 specifications, the system detects this and calculates the MSFT_Screen Coverage value of EXT_Lod in real-time to select the optimal LOD level. This can be expressed by the following equation:

$$LOD_{level} = f(W_{canas}, H_{canvas}), \quad (1)$$

where $\boldsymbol{f(canvas)}$ is a function that determines the optimal mapping between canvas size and LOD level. When the canvas size is reduced, the mesh details perceivable by the user are limited; thus, the system automatically selects a lower *c*omplexity LOD model to improve rendering efficiency.

3.3.2. Memory Usage-Based LOD Optimization

While 3D rendering in high-performance PC environments has become commonplace, memory resources remain limited on mobile devices such as smartphones. Considering these environmental constraints, this research has implemented a real-time memory monitoring mechanism. When the current memory usage of the browser approaches a threshold value, the MSFT_ScreenCoverage value is dynamically adjusted as follows:

$$MSFT_{ScreenCoverage} = \frac{\alpha(Mem_{max} - Mem_{current})}{Mem_{max}}, \quad (2)$$

where $\boldsymbol{\alpha}$ is an adjustment coefficient, and $\boldsymbol{Mem_{max}}$ c the maximum allowable memory usage in the system. This approach enables efficient visualization of 3D data even in memory-constrained environments.

3.3.3. User Direct Control Interface

Despite the convenience of automated LOD transition algorithms, there exist specialized use cases requiring analysis of mesh characteristics at specific LOD levels. To address this, our system integrates a hierarchical LOD Tree View UI, providing functionality for users to directly select their desired LOD level. This interface also displays metadata such as mesh complexity, vertex count, and polygon count for each LOD level, enabling informed selection.

The integration of these multiple LOD transition algorithms implements an adaptive rendering framework capable of flexibly responding to various hardware environments and user requirements, thereby simultaneously enhancing the efficiency of glTF model rendering in WebGL environments and the user experience.

IV. RESULT

Fig. 3 shows the results of the WebGL viewer developed using the technology proposed in this paper. For the experiment, we used high-resolution facial scan data. To apply each LOD level, we applied decimation values of 0.5, 0.09, 0.05, and 0.025 respectively to four stages based on the original mesh LOD0. The decimation process demonstrated effective resource reduction while minimally degrading the actual mesh quality.

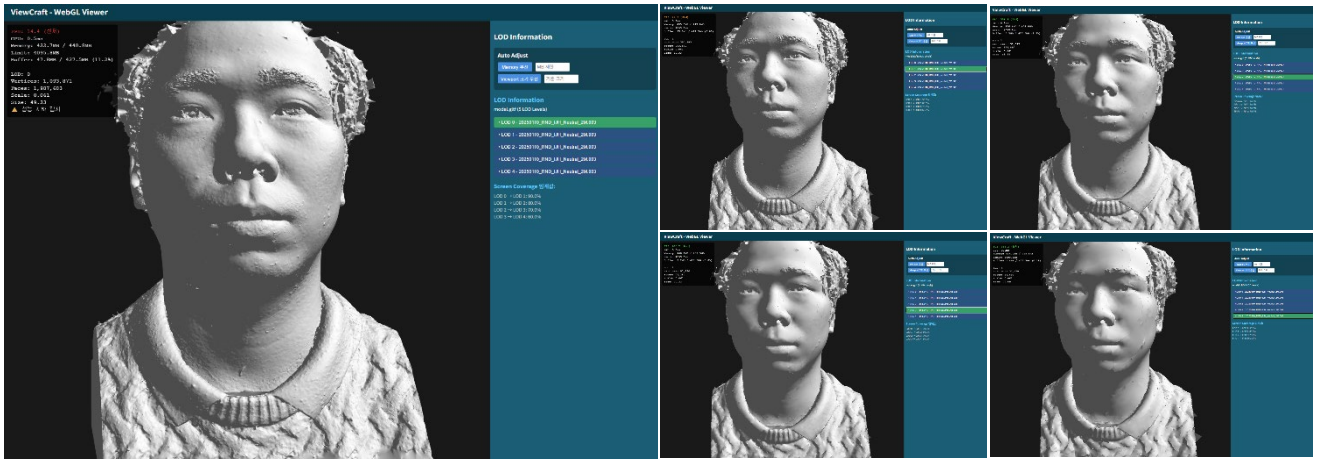


Fig. 3. Implementation of the proposed WebGL-based LOD viewer platform displaying a 3D scanned model with hierarchical LOD Tree View UI (right panel) and real-time performance metrics including FPS, memory usage, and screen coverage information.

We developed a system that not only provides functionality for users to directly select each LOD level to examine the mesh, but also enables automatic LOD application based on memory usage and screen canvas size. For user experience, the system is designed to activate only one mode among user direct selection, memory-based automatic selection, and screen size-based automatic selection, and provides customization functionality suited to user equipment performance by allowing users to directly set threshold memory and reference canvas size.

The performance metrics according to the LOD models used are shown in Table 1.

The reason why the actual memory used in the viewer is relatively small compared to the large glTF file size is as follows. The glTF models used in this research embedded high-resolution texture information, and due to the characteristics of high-resolution scan data, texture data was also stored in high resolution, increasing the overall file size. However, since our viewer focused on geometry information for LOD performance measurement and loaded only position and normal data into GPU buffers without loading texture information, we were able to optimize actual memory usage.

Table 1. Performance comparison of different LOD levels showing vertex count, file size, memory usage, and frame rate (FPS) for the proposed WebGL-based glTF viewer.

LOD	Vertices	File size (MB)	Memory usage (MB)	FPS
0	1,093,871	138	47.8	14.4
1	595,395	101	25.0	29.3
2	152,395	69	5.5	304.9
3	91,920	65	3.2	666.7
4	50,179	62	1.4	568.2

V. CONCLUSION AND DISCUSSION

In this research, we proposed a multi-LOD system for efficient glTF model rendering in WebGL environments. By developing a Python-based glTF Exporter independent of existing MSFT plugins, we implemented a platform-independent LOD integration environment while maintaining compatibility with existing formats through the utilization of EXT_Lod tags. Furthermore, through context-adaptive LOD transition algorithms, we implemented a rendering framework that can flexibly respond to various hardware environments and user requirements by providing three transition mechanisms: web browser canvas size, memory usage, and direct user control.

The system proposed in this research is based on several key design decisions. First, the adoption of an approach based on canvas size and memory usage, rather than traditional distance-based LOD transitions, stems from the observation that in most web-based 3D viewers, the distance between camera and object changes in a limited manner. In such environments, we considered that display environment and system resources have a more direct impact on user experience than distance.

Second, the development of a Python-based Exporter was a decision to enhance accessibility and scalability in the process of integrating glTF files generated from various 3D modeling software. In particular, it was designed to enable LOD integration even in environments where installing the MSFT Toolkit is difficult or where server-side processing is required. The Python-based exporter developed in this research was also implemented as a Blender plugin, enabling easy acquisition of data needed for experiments.

For future research, first, we aim to enhance the canvas size-based LOD algorithm by incorporating display pixel density (DPI) considerations, where high-DPI displays

could benefit from higher LOD levels even with smaller canvas sizes. Second, we aim to secure the possibility of extending the proposed LOD management technique beyond the WebGL environment to become an industry standard through the development of a glTF import system that can be utilized in major game engines such as Unreal Engine or Unity. Third, we plan to develop an integrated system that applies texture LOD alongside mesh LOD to further optimize glTF files, thereby optimizing overall system resource usage and improving rendering performance. Fourth, for models without explicitly defined LODs, we plan to develop a system capable of generating multiple LOD variations in real-time using Garland and Heckbert's Quadric Error Metrics [5] or recent deep learning-based mesh simplification methodologies [6].

Finally, we plan to expand the application range of web-based 3D rendering solutions through integration with blockchain technologies such as NFTs. By implementing ownership and access management of personalized 3D scan data with blockchain technology, we can provide a decentralized solution befitting the Web3.0 era, which is expected to present a new paradigm for digital asset management and sharing in metaverse environments.

ACKNOWLEDGEMENT

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2023-00229451, Interoperable Digital Human (Avatar) Interlocking Technology Between Heterogeneous Platforms).

REFERENCES

[1] T. Venturini, M. Brambilla, C. Celestini, A. Tarantino, and M. T. Grimaldi, "DECODE-3DViz: A web-based visualization system for large scale volumetric medical data," *Computers & Graphics*, vol. 91, pp. 76-89, 2020.

[2] Microsoft, *MSFT_LOD glTF Extension Specification*, Microsoft Corporation, 2018. https://github.com/microsoft/glTF/blob/master/extensions/2.0/Vendor/MSFT_lod/README.md.

[3] Cesium, *3D Tiles Specification*, AGI/Cesium, 2017. https://github.com/CesiumGS/3d-tiles.

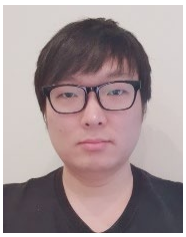
[4] Web3DP Project, *Web3DP: Open Platform for 3D Content Distribution Using NFT and Decentralized Storage*, Web3DP Whitepaper, 2021. https://web3dp.io

[5] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual Conference on Computer Graphics and Interactive Techniques*, 1997, pp. 209-216.

[6] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN: A network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1-12, 2019.

AUTHORS

Jaeho Im received his BS degree from Soongsil University and his MS and Ph.D. degrees from Korea University. He specializes in physics-based simulation, computer graphics, and virtual production systems, with extensive experience in visual effects (VFX). His research interests include advanced motion capture frameworks, monocular matchmoving calibration, and VR immersion techniques, including phobia treatments.
He has significantly contributed to real-time digital human production, facial animation through blendshape rig replication, and modular UI development for real-time platforms. His work involves complex simulations of smoke, fire, fluids, interactions between solids and fluids using adaptive signed distance fields, optimized algorithms for mobile devices, and realistic animations of wet hair and freezing water.



Minsoo Park received his MS degree in the Department of Civil and Environmental System Engineering from Seoul National University of Science and Technology, Korea, in 2016. His research interests include digital humans, databases, and the metaverse, especially in creating interactive spaces with realistic human representation.

Yujin Kim received her Bachelor's degree in Tourism Management from the National Institute for Lifelong Education, Korea in 2019. Her research interests include 3D rendering, real-time 3D visualization, and mesh simpli-fication.